WE CLAIM:

1. A computer-implemented method for producing a binary-level conditional branch reversal within a binary program on a computer architecture that supports a predicated execution, comprising:

obtaining a predicate expression representing a condition that influences a direction of program flow of the binary-level conditional branch to be reversed;

determining a binary-level transformation that causes the binary-level conditional branch to be triggered when an opposite condition is true; and

modifying the binary-level conditional branch with the determined binary-level transformation, wherein the binary-level conditional branch is reversed.

2. The computer-implemented method of claim 1, wherein obtaining the predicate expression comprises:

uniquely identifying predicates that influence the direction of program flow of the binary-level conditional branch to be reversed;

deducing relationships between the uniquely identified predicates; and

based on the relationships between the uniquely identified predicates, determining at least one predicate that influences the direction of program flow of the binary-level conditional branch.

3. The computer-implemented method of claim 2, further comprising locating speculative load instructions that impacts the truth value of a predicate associated with the binary-level conditional branch.

4. The computer-implemented method of claim 3, further comprising inserting into the binary program instructions to exclude execution of the binary-level conditional branch in response to a faulted speculative load.

5. The computer-implemented method of claim 2, wherein deducing the relationships includes conducting a predicate-aware, reaching definition data flow analysis.

6. The computer implemented method of claim 1, wherein determining the binary-level transformation comprises computing an inverse predicate expression that describes the opposite condition.

7. The computer implemented method of claim 6, further comprising determining whether at least one predicate in the inverse predicate expression is unmaterialized, and, if so, materializing the unmaterialized predicate.

8. The computer implemented method of claim 7, wherein materializing the unmaterialized predicate comprises:

locating a free register to support the unmaterialized predicate;

associating a new predicate with the free register; and

adding an instruction to define the new predicate as the unmaterialized predicate, wherein the unmaterialized predicate is now materialized.

9. The computer implemented method of claim 8, wherein locating the free register comprises conducting a predicate-aware liveness analysis.

10. The computer implemented method of claim 6, wherein if the inverse predicate expression includes multiple predicates, reducing the inverse predicate expression to a single predicate.

11. The computer implemented method of claim 10, wherein modifying the binary-level conditional branch comprises replacing an existing guarding predicate with the single predicate.

12. A computer-implemented method for obtaining a predicate expression that determines a guarding predicate of a binary-level conditional branch instruction within a binary program, comprising:

uniquely identifying predicates that influence a direction of program flow of the binary-level conditional branch to be reversed;

deducing relationships between the uniquely identified predicates; and

based on the relationships between the uniquely identified predicates, determining at least one predicate that influences the direction of program flow of the binary-level conditional branch.

13. The computer-implemented method of claim 12, further comprising locating a speculative load instruction that impacts the truth value of the guarding predicate associated with the binary-level conditional branch.

14. The computer-implemented method of claim 13, further comprising inserting into the binary program instructions to exclude execution of the binary-level conditional branch in response to a faulted speculative load.

15. The computer implemented method of claim 12, wherein deducing the relationships includes conducting a predicate-aware, reaching definition data flow analysis.

16. A computer-implemented method for determining a binary-level transformation that causes a binary-level conditional branch within a binary program to be triggered when an opposite condition is true, comprising computing an inverse predicate expression that describes the opposite condition.

17. The computer implemented method of claim 16, further comprising determining whether at least one predicate in the inverse predicate expression is unmaterialized, and, if so, materializing the unmaterialized predicate.

18. The computer implemented method of claim 17, wherein materializing the unmaterialized predicate comprises:

locating a free register to support the unmaterialized predicate;

associating a new predicate with the free register; and

adding an instruction to define the new predicate as the unmaterialized predicate, wherein the unmaterialized predicate is now materialized.

19. The computer implemented method of claim 18, wherein locating the free register comprises conducting a predicate-aware liveness analysis.

20. The computer implemented method of claim 16, wherein if the inverse predicate expression includes multiple predicates, reducing the inverse predicate expression to a single predicate.

21. The computer implemented method of claim 20, wherein modifying the binary-level conditional branch comprises replacing an existing guarding predicate with the single predicate.

22. A computer-readable medium having computer-executable instructions for producing a binary-level conditional branch reversal within a binary program on a computer architecture that supports a predicated execution, the instructions comprising:

obtaining a predicate expression representing a condition that influences a direction of program flow of the binary-level conditional branch to be reversed;

determining a binary-level transformation that causes the binary-level conditional branch to be triggered when an opposite condition is true; and

modifying the binary-level conditional branch with the determined binary-level transformation, wherein the binary-level conditional branch is reversed.